Lab 2 - report, Michał Błaszczak [245047]

Task 1

In [ ]:
```
import numpy as np
import pandas as pd
from datetime import datetime
```

In [ ]:
```
# removed first row from csv file because it caused importing one column only
# also renamed int columns
```

import data and set datetime index

In [ ]:
```
df_POL = pd.read_csv('multiTimeline.csv', index_col = 'Date', parse_dates = True)
df_USA = pd.read_csv('multiTimeline-2.csv', index_col = 'Date', parse_dates = True)
df_UK = pd.read_csv('multiTimeline-3.csv', index_col = 'Date', parse_dates = True)
```

In [ ]:
```
df_POL
```

Out[ ]:

	vacation: (Poland)
Date	
2004-01-01	42
2004-02-01	100
2004-03-01	68
2004-04-01	90
2004-05-01	62
...	...
2021-11-01	37
2021-12-01	69
2022-01-01	70
2022-02-01	66
2022-03-01	47

219 rows × 1 columns

rename columns

In [ ]:
```
df_POL = df_POL.rename(columns={'vacation: (Poland)': 'POL'})
df_UK = df_UK.rename(columns={'vacation: (United Kingdom)': 'UK'})
df_USA = df_USA.rename(columns={'vacation: (United States)': 'USA'})
```

combine dataframes into one

In [ ]:
```
df = pd.concat([df_POL, df_USA, df_UK], ignore_index=False, sort=False, axis=1)
df
```

Out[ ]:

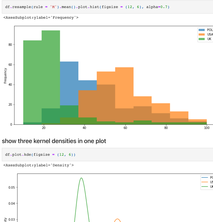
	POL	USA	UK
Date			
2004-01-01	42	87	38
2004-02-01	100	76	32
2004-03-01	68	71	35
2004-04-01	90	66	33
2004-05-01	62	60	33
...	...	...	...
2021-11-01	37	48	24
2021-12-01	69	61	42
2022-01-01	70	52	31
2022-02-01	66	53	27
2022-03-01	47	50	22

219 rows × 3 columns

plot the time series for all countries in one plot

In [ ]:
```
df_plot(figsize = (12, 6)).autoscale(axis = 'x', tight = True;
```

Out[ ]:



generate descriptive statistics

In [ ]:
```
df.info()
```

Out[ ]:
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 219 entries, 2004-01-01 to 2022-03-01
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   POL     219 non-null    int64
 1   USA     219 non-null    int64
 2   UK      219 non-null    int64
dtypes: int64(3)
memory usage: 6.8 KB
```

In [ ]:
```
df.describe()
```

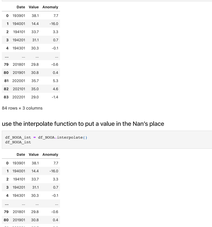
Out[ ]:

	POL	USA	UK
count	219.000000	219.000000	219.000000
mean	45.794521	58.301370	34.543379
std	17.214213	14.830266	14.441130
min	10.000000	26.000000	13.000000
25%	33.000000	49.000000	17.000000
50%	42.000000	56.000000	20.000000
75%	54.000000	67.000000	24.500000
max	100.000000	100.000000	100.000000

show three histograms in one plot

In [ ]:
```
df.resample(rule = 'M').mean().plot.hist(figsize = (12, 6), alpha=0.7)
```

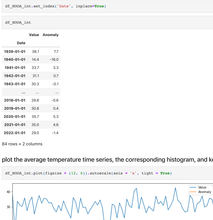
Out[ ]:
```
<AxesSubplot:ylabel='Frequency'>
```



show three kernel densities in one plot

In [ ]:
```
df.plot.kde(figsize = (12, 6))
```

Out[ ]:
```
<AxesSubplot:ylabel='Density'>
```



Task 2

import data

In [ ]:
```
# deleted few info rows because it caused problems when importing
df_NOAA = pd.read_csv('NOAA_df= Lista_Stazioni.csv')
df_NOAA
```

Out[ ]:

	Date	Value	Anomaly
0	193901	38.1	7.7
1	194001	14.4	-16.0
2	194101	33.7	3.3
3	194201	31.1	0.7
4	194301	30.3	-0.1
...	...	...	...
79	201901	29.8	-0.6
80	201901	30.8	0.4
81	202001	33.7	3.3
82	202101	35.0	4.6
83	202201	29.0	-1.4

84 rows × 3 columns

locate missing values and change them to nan

In [ ]:
```
df_NOAA.loc[ : , df_NOAA == -99] = np.nan
df_NOAA.to_of_NOAA(df_NOAA.head()
df_NOAA
```

Out[ ]:

	Date	Value	Anomaly
0	193901	38.1	7.7
1	194001	14.4	-16.0
2	194101	33.7	3.3
3	194201	31.1	0.7
4	194301	30.3	-0.1
...	...	...	...
79	201901	29.8	-0.6
80	201901	30.8	0.4
81	202001	33.7	3.3
82	202101	35.0	4.6
83	202201	29.0	-1.4

84 rows × 3 columns

use the interpolate function to put a value in the Nan's place

In [ ]:
```
df_NOAA_int = df_NOAA.interpolate()
df_NOAA_int
```

Out[ ]:

	Date	Value	Anomaly
0	193901	38.1	7.7
1	194001	14.4	-16.0
2	194101	33.7	3.3
3	194201	31.1	0.7
4	194301	30.3	-0.1
...	...	...	...
79	201901	29.8	-0.6
80	201901	30.8	0.4
81	202001	33.7	3.3
82	202101	35.0	4.6
83	202201	29.0	-1.4

84 rows × 3 columns

convert the index to datetime format

In [ ]:
```
df_NOAA_int['Date'] = pd.to_datetime(df_NOAA_int['Date'], format='%Y%m')
```

In [ ]:
```
df_NOAA_int.set_index('Date', inplace=True)
```

In [ ]:
```
df_NOAA_int
```

Out[ ]:

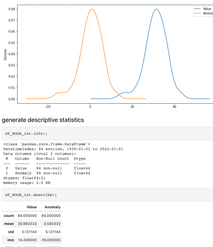
	Value	Anomaly
Date		
1939-01-01	38.1	7.7
1940-01-01	14.4	-16.0
1941-01-01	33.7	3.3
1942-01-01	31.1	0.7
1943-01-01	30.3	-0.1
...	...	...
2018-01-01	29.8	-0.6
2019-01-01	30.8	0.4
2020-01-01	33.7	3.3
2021-01-01	35.0	4.6
2022-01-01	29.0	-1.4

84 rows × 2 columns

plot the average temperature time series, the corresponding histogram, and kernel density plot

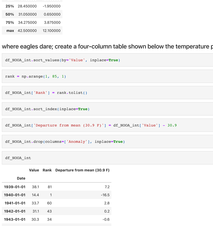
In [ ]:
```
df_NOAA_int.plot.figsize = (12, 6).autoscale(axis = 'x', tight = True)
```

Out[ ]:



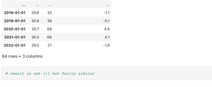
In [ ]:
```
df_NOAA_int.plot.hist(figsize = (12, 6), alpha=0.7)
```

Out[ ]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [ ]:
```
df_NOAA_int.plot.kde(figsize = (12, 6))
```

Out[ ]:
```
<AxesSubplot:ylabel='Density'>
```



generate descriptive statistics

In [ ]:
```
df_NOAA_int.info()
```

Out[ ]:
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84 entries, 1939-01-01 to 2022-01-01
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Value    84 non-null     float64
 1   Anomaly  84 non-null     float64
dtypes: float64(2)
memory usage: 2.0 KB
```

In [ ]:
```
df_NOAA_int.describe()
```

Out[ ]:

	Value	Anomaly
count	84.000000	84.000000
mean	30.883333	3.583333
std	5.131144	5.131144
min	14.400000	-16.000000
25%	28.450000	-1.950000
50%	31.050000	0.650000
75%	34.275000	3.875000
max	42.500000	12.100000

where eagles dare; create a four-column table shown below the temperature plot at the NOAA website

In [ ]:
```
df_NOAA_int.sort_values(by='Value', inplace=True)
```

In [ ]:
```
rank = np.arange(1, 85, 1)
```

In [ ]:
```
df_NOAA_int['Rank'] = rank.tolist()
```

In [ ]:
```
df_NOAA_int.sort_index(inplace=True)
```

In [ ]:
```
df_NOAA_int['departure from mean (30.9 F)'] = df_NOAA_int['Value'] - 30.9
```

In [ ]:
```
df_NOAA_int.drop(columns='Anomaly', inplace=True)
```

In [ ]:
```
df_NOAA_int
```

Out[ ]:

	Value	Rank	Departure from mean (30.9 F)
Date			
1939-01-01	38.1	81	7.2
1940-01-01	14.4	1	-16.5
1941-01-01	33.7	60	2.8
1942-01-01	31.1	43	0.2
1943-01-01	30.3	34	-0.6
...	...	...	...
2018-01-01	29.8	33	-1.1
2019-01-01	30.8	38	-0.1
2020-01-01	33.7	69	4.8
2021-01-01	35.0	66	4.1
2022-01-01	29.0	27	-1.9

84 rows × 3 columns

In [ ]:
```
# result is not (!) but fairly similar
```